## Turing Machine Computation History Verification by a PDA

Let $M = (Q, \Sigma, \Gamma, \delta_M, q_s, q_a, q_r)$ be a deterministic Turing machine and $w$ be an input to $M$. Let $\Delta = Q \cup \Gamma \cup \{\#\}$. Assume that $\# \notin Q$ and $\# \notin \Gamma$. Given a string $U\#V \in \Delta^*$, we design a PDA $P$ that accepts iff $V$ is not a valid immediate successor configuration of $U$ on input $w$.

If $U$ or $V$ is not a valid configuration of $M$ on $w$ then $P$ accepts. (A DFA can be used to test that a string $U$ (or $V$) is not a valid configuration of $M$ on $w$.)

Suppose $U$ and $V$ are valid configurations. Let $U = u_1u_2 \dots u_r$ and $V = v_1v_2 \dots v_s$. For $i \geq 1$, consider the $2 \times 3$ windows: $\left[\begin{array}{c|c|c} u_i & u_{i+1} & u_{i+2} \\ \hline v_i & v_{i+1} & v_{i+2} \end{array}\right]$.

The idea is to consider the windows one by one and nondeterministically do either: (a) compare the first symbols of the two rows and accept if they differ; or (b) go to the next window. In doing this, we have to take care of the possibility that the symbols around the state may differ (as dictated by the transition function $\delta_M$). To take care of this situation we define a window to be a *critical window* if $u_{i+1}$ is the state symbol in $U$ and process critical windows differently. If $u_1$ is the state symbol in $U$ then the first $2 \times 2$ window $\left[\begin{array}{c|c} u_1 & u_2 \\ \hline v_1 & v_2 \end{array}\right]$ is called the *critical window*. Note that there is only one critical window.

Note that in the description below it is enough to push some symbol $X$ onto the stack (since the stack is used to get to the right position in $V$).

The description of the PDA $P$ below uses a procedure COMPARE($CurrentSymbol, i$) that accepts iff $CurrentSymbol \neq v_i$. The procedure COMPARE is described later. We will assume that the stack has a bottom marker, say, $Z$.

STEP 1: Let $u_1 \in Q$. (R/W head is pointing to the first symbol of the tape of $M$. The first $2 \times 2$ window is the critical window.)

Let $\delta_M(u_1, u_2) = (p, a, L)$. (The correct critical window is $\left[\begin{array}{c|c} u_1 & u_2 \\ \hline p & a \end{array}\right]$. Handle the Right move similarly.)

Nondeterministically do one of the following two actions:

1. $CurrentSymbol = p$; COMPARE($CurrentSymbol, 1$);
2. Push $p$ onto the Stack; Nondeterministically do one of the following two actions:
   (a) $CurrentSymbol = a$; COMPARE($CurrentSymbol, 2$);
   (b) Push $a$ onto the Stack; $i = 3$; Go to Step 3 to process the post-critical-window part of $U$;

STEP 2: Let $u_1 \notin Q$; (R/W head is not pointing to the first symbol of the tape of $M$. Processing takes place in three stages:(a) pre-critical-window, (b) critical-window, and (c) post-critical-window.)

$i = 1$;

STEP 2.1: (pre-critical-window stage)

REPEAT as long as $u_{i+1} \notin Q$ (pre-critical-window stage)

Non-deterministically do one of the following two actions:

1. $CurrentSymbol = u_i$; COMPARE($CurrentSymbol, i$);
2. Push $CurrentSymbol$ onto the stack; $i = i + 1$;

STEP 2.2: ($u_{i+1} \in Q$: critical-window stage)

Let $\delta_M(u_{i+1}, u_{i+2}) = (p, a, L)$. (The correct critical-window is: $\left[\begin{array}{c|c|c} u_i & u_{i+1} & u_{i+2} \\ \hline p & u_i & a \end{array}\right]$. Handle the Right and Stationary moves similarly.)

Non-deterministically do one of the following two actions:

1. $CurrentSymbol = p$; COMPARE($CurrentSymbol, i$);
2. Push $p$ onto the Stack; $i = i + 1$; Nondeterministically do one of the following two actions:
   (a) $CurrentSymbol = u_{i-1}$; COMPARE($CurrentSymbol, i$);
   (b) Push $u_{i-1}$ onto the Stack; $i = i + 1$; Non-deterministically do one of the following two actions:
      i. $CurrentSymbol = a$; COMPARE($CurrentSymbol, i$);
      ii. Push $a$ onto the Stack; $i = i + 1$; Go to Step 3 to process the post-critical-window part of $U$;

STEP 3: (Post-critical-window stage)

REPEAT as long as ($u_i \neq \#$):

Non-deterministically do one of the following two actions:

1. $CurrentSymbol = u_i$; COMPARE($CurrentSymbol, i$);
2. Push $u_i$ onto the Stack; $i = i + 1$;

($u_i = \#$): $CurrentSymbol = \#$; COMPARE($CurrentSymbol, i$);

Procedure COMPARE($CurrentSymbol, i$).
(Accept iff $CurrentSymbol \neq v_i$; The stack has $i - 1$ symbols.)

IF ($CurrentSymbol \neq \#$) THEN Read and ignore symbols until and including the # mark;

(The Read head of the PDA should now be pointing to the first symbol of $V$.)

WHILE ($Stacktop \neq Z$) DO: Read the next symbol of $V$ and pop the stack;

(The Read head of the PDA should be pointing to $v_i$.)

IF $v_i = CurrentSymbol$ THEN reject ELSE accept.